Milestone 2 Report

1 业务需求

1.1 背景与机遇

随着计算机技术的发展，开源软件在操作系统、编译工具链、数据库、WEB服务器、移动操作系统等各个方面已经成为主流。开源项目是指由开源贡献者发起的遵守指定开源协议的开放源代码的项目。开发者可以在 GitHub 等开源平台去浏览感兴趣的开源项目，随时可以了解和参与任何一个开源项目，甚至成为其中的核心贡献者。开源项目是开源世界的未来，是未来所有的可能性。

然而，开源需要付出很多精力去维护，由于投入精力的不足，往往会导致项目胎死腹中的情况。而且，开源项目获取关注度的方式有限，一个项目从孵化到使用，经历的时间往往会比较长，加上工作生活上的压力，项目可能会因此戛然而止。开源项目回报率太低，物质方面收入少，因此，项目可能因为资金问题而折戟。开源项目没有明确规划,或者开源组织对项目的信心不足，都会导致项目的遗憾中断。

开源项目的维护方式，除了个人维护，还可以由开源社区进行维护。社区围绕着一个共同的目标，将一个或者若干个项目向前推进。社区针对项目的投入不光体现在对项目功能的开发和 Bug 修正，更是从更高的层面来提升项目的影响力、路线图、沟通协作和质量保证等。项目的影响力主要体现在使用者的质量和数量。

为了更好的发展开源软件，需要对其有深入的分析和了解。然而开源软件开发过程中的协作程度、开发者的贡献程度、项目的影响力等开源平台等并没有提供直接的数据，需要进行进一步的分析。因此开源软件分析平台可以通过分析git的数据和一些开源平台提供的数据来帮助维护者和管理者以及投资方深入分析开源软件，帮助评估项目的成果，并且深入分析团队开发流程和协作，根据数据清晰的了解软件开发中的软件稳定性、团队弹性等。

1.2 业务风险

- github网页访问不稳定
- 本地缓存过多，占用内存过大

2 项目前景

2.1 概述

本项目作为2021–2022学年”软件需求工程“课程第2组大作业“开源项目分析平台”的迭代开发，为实现分析开源项目、、评估开源成果这一业务需求，达到完成2022–2023学年”软件需求工程“课程第6组大作业的业务目标，在本次迭代中，实现下列特性有较高优先级。

2.2 主要特性

FE–1 优化数据获取，实现数据源缓存

FE–2 细化数据过滤，提高对多样化仓库的适配

FE–3 优化从 GitHub 上获取信息的方式，在 GitHub 接口不可用时仍能提供项目历史数据访问

FE–4 重新设计数据库模型

FE–5 更换前端框架，实现前后端分离开发

FE–6 统计一段时间内来自不同贡献者的代码提交数，指出项目的核心贡献者

FE–7 取得 commit 信息，筛选出设计相关的讨论

FE–8 以周为单位，可视化展示设计讨论数量随时间的变化

FE–9 优化前端展示方式和界面设计

FE–10 定义 code smell并衡量最新版代码设计质量

FE–11 以 PyTorch 项目为例，进行数据可视化

FE–11–1 查询stargazer, committer, issue 信息，按company组织数据

FE–11–2 计算占比并将其可视化，在可视化效果上体现出占比大小等不同

FE–11–3 进一步，按内部不同模块进行上述处理

2.3 假设与依赖

AS–1 平台可以调用Github API，得到数据库中待检索仓库的全部提交信息和贡献者信息。

AS–2 后端可以得到足够的服务器支持，用来进行训练神经网络分析提交信息和贡献程度。

AS–3 平台可以稳定连接数据库。

3 项目范围

3.1 版本范围

主要特性	第一轮迭代	第二轮迭代
FE–1 优化数据获取，实现数据源缓存	实现	
FE–2 细化数据过滤，提高对多样化仓库的适配	实现	
FE–3 优化从 GitHub 上获取信息的方式，在 GitHub 接口不可用时仍能提供项目历史数据访问	实现	
FE–4 重新设计数据库模型	实现	
FE–5 更换前端框架，实现前后端分离开发	实现	
FE–6 统计一段时间内来自不同贡献者的代码提交数，指出项目的核心贡献者	实现	
FE–7 取得 commit 信息，筛选出设计相关的讨论		实现
FE–8 一周为单位，可视化展示设计讨论数量随时间的变化		实现
FE–9 优化前端展示方式和界面设计	实现	
FE–10 定义 code smell并衡量最新版代码设计质量		实现
FE–11 以 PyTorch 项目为例，进行数据可视化		实现
FE–11–1 查询stargazer, committer, issue 信息，按company组织数据		实现
FE–11–2 计算占比并将其可视化，在可视化效果上体现出占比大小等不同		实现
FE–11–3 进一步，按内部不同模块进行上述处理		实现

3.2 限制与排除

实现范围

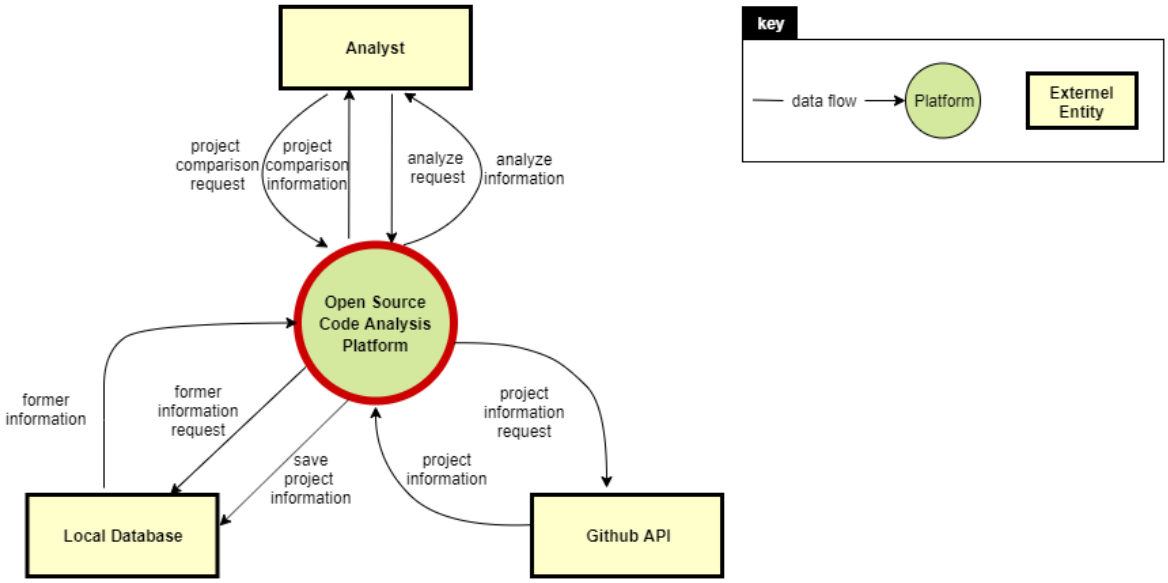
- 统计一段时间内来自不同贡献者的代码提交数，指出项目的核心贡献者
- 对 PyTorch 项目的 stargazer, committer, issue 人数的 company 信息分别进行数据
    - 计算占比并将其可视化，在可视化效果上体现出占比大小等不同
    - 可以计算该项目内部不同 module（可以根据文件目录进行分块）的公司占比
- 优化数据获取
    - 实现针对数据源的缓存，将数据与实际数据来源解耦，降低由于网络问题可能引发的风险
    - 细化数据过滤，对可能缺失的数据进行特殊处理，提升用户友好程度
    - 优化从 GitHub 上获取信息的方式，在 GitHub 接口不可用时仍能提供项目历史数据访问
- 进行项目重构与满足迭代需求
    - 让其界面友好、美观
    - 实现URL跳转项目代码
    - 增加关键注释
    - 完善README 文档

无需实现范围

- 项目设计讨论数据收集
    - 筛选出设计相关的讨论，筛选标准可以参考下一页图中的设计话题类别及描述
    - 基于上条，可视化设计讨论的数量是如何随时间变化的，以周为单位
    - 使用 code smell 衡量最新版代码设计质量，基于 InFusion 定义 code smell
- 非功能性需求
    - 优化加载速度
    - 提高并发

4 上下文图(context diagram)

Context Diagram



5 路线图(roadmap)